WEST

Freeform Search

Database: US Patents Full-Text Database
US Pre-Grant Publication Full-Text Database
JPO Abstracts Database
EPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

Term: l1 and (plurality with units)

Display: 10 **Documents in Display Format:** KWIC **Starting with Number** 1

Generate: ○ **Hit List** ◉ **Hit Count** ○ **Image**

Search | Clear | Help | Logout | Interrupt

Main Menu | Show S Numbers | Edit S Numbers | Preferences

Search History

Today's Date: 11/14/2001

DB Name	Query	Hit Count	Set Name
USPT	l1 and (plurality with units)	6	L4
USPT	l1 and (plurality with associat$ with units)	2	L3
USPT	l1 and (plurality with functional with units)	2	L2
USPT	(register$ with partition$ with global)	21	L1

L1: Entry 19 of 21 File: USPT Jan 12, 1993

DOCUMENT-IDENTIFIER: US 5179681 A
TITLE: Method and apparatus for current window cache with switchable address and out cache registers

BSPR:
The general registers include from forty to five hundred twenty 32 bit registers. Whatever the total number of general registers, these registers are partitioned into eight global registers and a number of sixteen registers sets, each set divided into eight IN and eight local registers. At any time, an instruction can access a window including the eight global registers, the IN and local registers of one set of registers, and the IN registers of a logically-adjacent set of registers. These IN registers of the logically-adjacent set of registers are addressed as the OUT registers of the sixteen register set of the window including both IN and local registers. Thus, an instruction can access a window including the eight global registers, the IN and local registers of one set of registers, and the IN registers addressed as OUT registers of the logically-adjacent set of registers.

WEST

Generate Collection

L1: Entry 18 of 21 File: USPT Jan 12, 1993

DOCUMENT-IDENTIFIER: US 5179682 A
TITLE: Method and apparatus for improved current window cache with switchable address IN, OUT, and local cache registers

BSPR:
The general purpose registers include from forty to five hundred and twenty 32 bit registers. Whatever the total number of general registers, these registers are partitioned into eight global registers and a number of sixteen registers sets, each set divided into eight IN and eight local registers. At any time, an instruction can access a window including the eight global registers, the IN and local registers of one set of registers, and the IN registers of a logically-adjacent set of registers. These IN registers of the logically-adjacent set of registers are addressed as the OUT registers of the sixteen register set of the window including both IN and local registers. Thus, an instruction can access a window including the eight global registers, the IN and local registers of one set of registers, and the IN registers (addressed as OUT registers) of the logically adjacent set of registers.

L1: Entry 11 of 21 File: USPT Mar 30, 1999

DOCUMENT-IDENTIFIER: US 5890000 A
TITLE: Cooperation of global and local register allocators for better handling of procedures

ABPL:
A method and device for optimizing a compiler involves cooperation between the global and local register allocators in assigning symbolic registers to hardware registers. A large procedure may have many associated symbolic registers; the invention involves partitioning the symbolic registers into at least two portions, and allowing the global register allocator to assign one portion and the local register allocator to assign another portion. The registers may be partitioned based on different criteria, such as local vs. global registers, or spill costs, or shallow vs. nested regions.

DEPR:
As shown in FIG. 1, the method involves mapping an arbitrary number of symbolic register sets, such as a set of symbolic registers for a basic block, onto a finite set of hardware registers 14 in a computational device (ie., a computer processor). The symbolic register sets correspond to an intermediate representation 12 of source code. The method requires partitioning at least one of the symbolic register sets into a plurality of portions (one or more symbolic registers) as indicated at 10a, 10b, 10c, 10d, assigning one of the hardware registers to at least one portion using a first register allocator, such as the local register allocator, and assigning the same or a different hardware register to at least one other portion using a second register allocator, such as the global register allocator. In this manner, any particularly large procedure in the object code is optimized with respect to compile time and efficiency of the object code. A global register allocator generally produces more efficient code, particularly for symbolic registers having global live ranges, but it is more complex and requires greater compile time. A local register allocator saves on compile time but generates less efficient code.

DEPR:
There are many different kinds of register allocators used in different optimizing compilers, so the present invention should not be construed as being specifically limited to cooperation between the local and global register allocators. For example, a compiler might also have an intermediate or medium-range register allocator (using a hybrid approach in order to be more compile-time efficient, but generate slightly less efficient code), and the present invention could be carried out by using any combination of these three register allocators to allocate the partitioned portions of the symbolic register set (i.e., using only the local and global allocators, or only the local and intermediate allocators, or only the intermediate and global allocators, or using all three). For those systems having local and global allocators, the local register allocator can assign the first portion of the symbolic register, and the global allocator assigns the rest. This ordering of the execution of the allocators should not be considered limiting to the scope of the invention. The allocators could execute in any order, both with respect to each other and with respect to other optimization phases of the compiler.

DEPR:
For the local and global allocator cooperation, the registers may be partitioned in many different ways. One method is shown in FIG. 3. A symbolic register in the set is examined (24,26) according to a selected criterion. Depending on what order the register assignment is to be done, the global register allocator may mark a symbolic register for subsequent assignment by the local register allocator (as illustrated by steps 28 and 30 in FIG. 3), or the local register allocator may mark a symbolic register for subsequent assignment by the global register allocator (not shown). This process is repeated for each symbolic register in the set (32,34).

L1: Entry 15 of 21 File: USPT Nov 22, 1994

DOCUMENT-IDENTIFIER: US 5367651 A
TITLE: Integrated register allocation, instruction scheduling, instruction reduction and loop unrolling

ABPL:
An improved register allocator, an improved instruction scheduler, an instruction combiner, and an improved loop unroller is provided to the code generator of a compiler of a computer system. Both the improved instruction scheduler and the improved loop unroller support a "preliminary" and a "final" mode of operation. Upon invocation, the improved register allocator determines and prioritizes regions of the program being compiled. Next, the improved register allocator, in cooperation with the improved instruction scheduler, the instruction combiner, and the improved loop unroller, determines the optimal partitioning for global and local registers for each region. Then, the improved register allocator allocates registers to each region based on the determined number of global registers for the region. After allocating registers for each region, the improved register allocator merges the regions together. The improved loop unroller and the improved instruction scheduler are then invoked successively in "final" mode to unroll the various loops and schedule the instructions being generated.

BSPR:
The advantageous results are achieved by providing an improved register allocator, an improved instruction scheduler, an instruction combiner, and an improved loop unroller to the code generator of a compiler of the target machine. Both the improved instruction scheduler and the improved loop unroller support a "preliminary" and a "final" mode of operation. Upon invocation, the improved register allocator determines and prioritizes regions of the program being compiled. Next, the improved register allocator, in cooperation with the improved instruction scheduler, the instruction combiner, and the improved loop unroller, determines the optimal partitioning for global and local registers for each region. Then, the improved register allocator allocates registers to each region based on the determined number of global registers for the region. After allocating registers for each region, the improved register allocator merges the regions together. The improved loop unroller and the improved instruction scheduler are then invoked successively in "final" mode to unroll the various loops and schedule the instructions being generated.

BSPR:
The improved register allocator determines the optimal global and local register partitions for a region by first making a preliminary register allocation for the region using all registers of the target machine. Upon making the preliminary allocation, the improved register allocator invokes the improved instruction scheduler in "preliminary" mode. In response, the improved instruction scheduler provides the improved register allocator with a preliminary scheduling of the instructions being generated. The improved instruction scheduler makes a preliminary instruction schedule using a copy of all registers in the target machine in addition to the globally allocated registers. The improved register allocator determines the optimal global/local register partition based on whether the improved register allocator or the improved instruction scheduler can better utilize a given register.

DRPR:
FIG. 9 illustrates an exemplary global and local register partitioning under the present invention.

DEPR:
The improved register allocator 48 receives the optimized intermediate representations and associated information as inputs. In response, the improved register allocator 48, in cooperation with the improved instruction scheduler, the

improved loop unroller and the instruction combiner of the present invention, determines the optimal partitioning for global and local registers for each region of the program, allocates the global registers of each region and then merges the regions together.

DEPR:
Then the improved register allocator determines the optimal partition for global and local registers, block 74. The improved register allocator determines the optimal partition by determining whether the improved register allocator or the improved scheduler can better utilize a particular register. For simplicity, in the presently preferred embodiment, the improved register allocator determines the utility based on usage and definition counts. It will be appreciated that the improved register allocator may be implemented with a variety of other utility determination functions.

DEPR:
Upon determining the optimal partition for global and local registers for the region, the improved register allocator makes a final allocation of the global registers, block 76.

DEPR:
The process, blocks 62-76, is repeated for each region in order of their priorities, until the optimal partition for global and local registers is determined for each region, and the global registers of each region are allocated. The global register allocation for the various regions are then merged together, block 80, introducing spill code if necessary. Merging is a process whereby situations like a register candidate is assigned to different registers in different regions or assigned to a register in one region and spilled in another are reconciled.

DEPR:
Referring now to FIG. 9, a block diagram illustrating an exemplary global and local register partitioning is shown. Shown is a program 92 comprising three basic blocks, 94a-94c. Block 1 94a comprises a load instruction having a register candidate REG A that is live through block 2 94b and block 3 94c. Block 2 94b comprises two groups of Load, Load, Add and Store instructions involving register candidates REG B through REG G that are live only within block 2 94b. Lastly, block 3 94c comprises a group of Load, Add and Store instructions involving register candidates REG A, REG H and REG I. REG H and REG I are both live within block 3 94c only.

CLPV:
d) determining optimal partitioning of said registers of said computer system into global and local registers, said partitioning determination being made based on the relative utilities of each of said registers to global register allocation and instruction scheduling;

CLPV:
e) allocating global registers to said instructions being generated for a second time using said registers of said computer system partitioned to be global registers;

CLPV:
d) partition determination means coupled to said first register allocation for determining optimal partitioning of said registers of said computer system into global and local registers, said partitioning determination being made based on the relative utility of each of said registers to global register allocation and instruction scheduling;

CLPV:
e) second register allocation means coupled to said first register allocation means for allocating global registers to said instructions being generated for a second time using said registers of said computer system partitioned to be global registers;

CLPV:
f) partition determinator coupled to said first register allocator for determining optimal partitioning of said registers of said computer system into global and local registers, said partitioning determination being made based on the relative

utility of each of said registers to global register allocation and instruction scheduling;

CLPV:
g) second register allocator coupled to said first register allocator for allocating global registers to said instructions being generated for a second time using said registers of said computer system partitioned to be global registers;

CLPW:
c.4) a partition determinator coupled to said first register allocator for determining optimal partitioning of said registers of said computer system into global and local registers, said partitioning determination being made based on the relative utility of each of said registers to global register allocation and instruction scheduling;

CLPW:
c.5) a second register allocator coupled to said first register allocator for allocating global registers to said instructions being generated for a second time using said registers of said computer system partitioned to be global registers;

CLPW:
c.4) a partition determinator coupled to said first register allocator for determining optimal partitioning of said registers of said computer system into global and local registers, said partitioning determination being made based on the relative utility of each of said registers to global register allocation and instruction scheduling;

CLPW:
c.5) a second register allocator coupled to said first register allocator for allocating global registers to said instructions being generated for a second time using said registers of said computer system partitioned to be global registers;